Ministry of Education

System Development Life Cycle (SDLC) methodologies – BPP Statement of Direction

Version 1.0

September 25, 2015

Information Security Classification : Low

Table of Contents

Introduction

The purpose of this document is to provide Ministry BPP's position (Statement of Direction) on the usage of various System Development Life Cycles (SDLCs) by Ministry projects.

Related References

- "*Iterative Development Standards*" available on BPP Sharepoint site.

Assumptions

Ministry of Education's Business Program Planning (BPP) process is oriented towards "*Waterfall*" approach to a large extent and towards "*Iterative and Incremental*" approach to some extent. Other approaches such as "Agile development" in strict sense are not currently supported by the BPP.

Scope

Ministry BPP's position (Statement of Direction) on the usage of various System Development Life Cycle (SDLC) methodologies described in this document is applicable to all types of Ministry projects of any size - *Custom development*, *COTS*, *Open-Source implementation*, *Software as a Service (SaaS)*, *Github* projects etc. **Out of scope :** Evaluations of the SDLC methodologies (pros and cons etc) and recommendations on any particular methodology and its process/usage etc is out of scope for this document. Some minimal amount of description on SDLC methodologies is provided only for context purposes.

Projects need to obtain prior approval from Ministry Architecture Committee (MAC) for using any different SDLC methodology than the "Waterfall" approach.

Objectives

The objectives of this document are to provide Ministry BPP's position (Statement of Direction) on the usage of various System Development Life Cycle (SDLC) methodologies as well as to briefly describe BPP's expectations from projects on the various SDLC methodologies used. The document also attempts to highlight some possible impacts on various areas and processes in Ministry (ITMB and Program Areas) when a project does not follow "Waterfall" approach for software development. Impacted areas in Ministry may incorporate any changes required to their processes to address the impacts due to the project's non-Waterfall approach to software development OR notify the area's concerns (if any) to the BPP team or ITMB.

Impacts

Not following "Waterfall" approach may create impacts on various ITMB/BPP processes such as "*Team Reviews and approvals/signoff*" process, *Change Management/release management* process etc and also on client/program area processes such as *UAT Testing* process. There could also be impacts on financial processes (Budget forecasts/estimates, costing and planning etc). Hence, if a project is not going to follow "Waterfall" model for strong business and/or technical reasons, then the project needs to notify to Ministry (BPP team or ITMB) well in advance (preferably at *pre-project* phase) with brief reasons and rationale in order to obtain prior approval from Ministry Architecture Committee (MAC) on the SDLC approach.

Impacts on Financial processes

In the event of project not following "Waterfall" method, various financial planning tasks such as budget forecasts/estimates, project costing estimates etc may be more complicated due to granularity of project variables such as time, efforts, deliverables, contracted resources etc. Also, in the Iterative approaches, the full business requirements may not be clear until the final iteration is completed leading to uncertainties in the actual requirements and the cost estimates for the implementation of those requirements.

Impacts on Project Management

In the event of project not following "Waterfall" method, the tracking of project progress may become more granular - due to project level management needed as well as iteration/increment level management needed for that project. The project must consult with the Stakeholders on the level at which project progress is to be reported to them – e.g. progress for each iteration or for a group of logical iterations etc.

Impacts on Clients/Program Areas

If "Business Requirements" phase is also to follow the Iterative approach, then there may be impacts on Clients/Program Areas. Clients/program areas may need to provide requirement inputs and approvals/signoffs for the BRD multiple times (i.e. for each iteration). Also, the client's UAT testing team will need to do UAT testing of each iteration when it is ready for release, as opposed to doing UAT testing of whole system in waterfall approach.

Impacts on Ministry (ITMB) review process

There will be more frequent team reviews (QCIL reviews) of a deliverable due to iterations of the same deliverable coming for review frequently. However, the content of the iteration deliverable to be reviewed is expected to be smaller compared to content of a full deliverable of whole application.

The Ministry Project Delivery Office (PDO) will have to do more frequent QCIL review initiations and follow ups due to iterations of the same deliverable coming for QCIL review frequently.

Impacts on Management of Deliverables/artefacts

There could be more deliverables with multiple artefacts (documents, models, diagrams etc) pertaining to each iteration. Versioning of all such smaller "iteration deliverables" and their ultimate consolidation or linking could pose challenges, unless a single master deliverable is contemplated with color coding etc. , distinguishing the iteration's content from the previous iteration. Also, the repository/container (such as Sharepoint site, Subversion etc for holding the documents/deliverables/artefacts) will have more granular content for a given application.

Impacts on Change Management/Release Management

There could be more number of Change Requests (CRs) and Release Management requests coming for release pertaining to various iterations, which in turn could result in more loads on the process and on the tools and repositories such as Subversion, Anthill, C2 etc.

Impacts on Testing

Development teams need to do progressive integration of iterations throughout the system development and also need to do progressive integration testing of iterations to avoid any cross-iteration issues/bugs. A final "whole application" testing is also recommended after the last iteration is rolled out.

There may be impacts on UAT testing and Performance (Stress/Load) testing processes due to frequent iterations coming for testing. UAT teams (in the client/program areas) will need to do UAT testing of each iteration when it is ready for release, as opposed to doing testing of whole system/application in waterfall approach.

In case of performance testing, there could be need to do a final performance testing of the "whole application" at the end , for the full application comprising of all iterations released, in order to get a more realistic performance test results from the whole application.

System Development Life Cycle (SDLC)

A system development life cycle (SDLC) is composed of a number of clearly defined and distinct work phases which are used by systems engineers and systems developers to plan for, design, build, test, and deliver information systems. An SDLC aims to produce high quality systems that meet or exceed user/customer expectations, based on user/customer requirements, by delivering systems which move through each clearly defined phase, within scheduled time-frames and cost estimates. Software development is a complex process and may also involve linking/interfacing of multiple systems (particularly in SOA scenario). To manage the level of complexity, a number of SDLC models or methodologies are available, such as : "*Waterfall*", "*Iterative and Incremental*", "*Agile software development*", "*Rapid prototyping*", "*synchronize and stabilize*", "*spiral*" etc.

Ministry of Education's Business Program Planning (BPP) process is mostly oriented towards "Waterfall" model and to some extent towards "*Iterative and Incremental*" model. Not following "Waterfall" model creates impacts on various BPP processes such as "Team Reviews and approvals/signoff" process, Change Management/release management process etc. Hence, if a project is not going to follow "Waterfall" model for strong business and/or technical reasons, then the project needs to notify to Ministry (BPP team or ITMB) well in advance (preferably at *pre-project* phase) with brief reasons and rationale.

BPP's expectations from projects on the various SDLC models are summarized below.

Waterfall method

All BPP deliverables from all phases will typically have one cycle for *team review* (QCIL review), one cycle for *issue-resolution* and one cycle for *deliverable approvals/signoff*s. The duration of *team review* cycle is typically 3-5 business days. The duration of *issue-resolution* cycle will depend on the time taken by the project to provide resolutions to feedback, which in turn may depend on the amount of feedback received from team review. In exceptional cases the "team review" and "issue-resolution" may go through multiple cycles (2 or 3 cycles max).

Iterative and Incremental method

The basic idea behind the "Iterative" model is to develop a system through repeated cycles (iterations) and release in smaller "increments" at a time, allowing software developers to take advantage of what was learned during development of earlier iterations/increments of the system. Lessons learnt comes from both the development and use of the system.

It is assumed that *Pre-project* phase (Business case, decision making, feasibility assessment and Options Analysis etc) and most of *Project Management* phase work is done for the project as a whole and not repeated for each

iteration. Some amount of "Project Management" phase work (called "Project Control List" in Iterative paradigm) may be needed for each iteration. Total number of iterations anticipated in the project and a brief scope of each iteration (eg : what functionality will be delivered by the iteration) needs to be communicated to Ministry well in advance to help in their resources planning for team reviews, UAT testing etc. Number of iterations should be kept as low as possible (typically 2-3 iterations max) to avoid too much granularity and any overload on processes, resources and tools/repositories due to number of iterations.

The "*Requirements*", "*Architecture / Design*", "*Testing*", "*User Acceptance*", "*Production rollout*", "*Post-iteration analysis and lessons learnt*" phases are repeated for each iteration. BPP deliverables are submitted by the projects for team review for each iteration. Each iteration deliverable needs to clearly identify (through color coding, labelling etc) the content of the iteration that is added over the previous iteration and that is to be reviewed exclusively for the given iteration. The duration of *team review* cycle for the iteration's deliverable review may be reduced to 2-3 days if feasible, due to possibly smaller size of the deliverable content to be reviewed.

Agile Software development method

"Agile software development" refers to a group of software development methodologies based on iterative development, where requirements and solutions evolve via *collaboration between self-organizing cross-functional teams*. Agile software development uses iterative development as a basis but advocates a lighter and more people-centric viewpoint than traditional approaches. Agile processes fundamentally incorporate iterations (of typically *1-6 weeks* duration) and the continuous feedback that it provides to successively refine and deliver a software system. One of the differences between agile and waterfall is the approach to quality and testing. In the waterfall model, there is always a separate *testing phase* after a build phase; however, in agile development, testing is usually done concurrently with or at least in the same iteration as the programming/coding. The agile approach also introduces a "***product***" mindset rather than the waterfall model's "***project***" mindset.

A typical high-level manifesto for Agile Software Development is :

Individuals and interactions over Processes and tools
Working software over Comprehensive documentation
Customer collaboration over Contract negotiation
Responding to change over Following a plan

Agile Software Development approach in strict sense is not currently supported by the Ministry BPP. In order to support the Agile Software Development approach, there is need to first address concerns such as how to handle the security, compliance, team reviews, quality control, documentation, project management, application audit requirements, knowledge retention and knowledge transfer etc in Agile paradigm – all of which need formal documentation of software at various phases of software development, which is contrary to the actual Agile manifesto.

Typical SDLC Phases

Regardless of which SDLC model is followed for any given project, some or all of the following phases/disciplines are involved in most types of projects (Custom development, COTS, Open-Source implementation, Software as a Service (SaaS), Github projects etc). In case of iterative/incremental approaches, the phases/disciplines are typically repeated for each iteration/increment. Work done in some phases/disciplines may be common and applicable to the whole project (to all iterations and increments of the project).

Planning : Pre-project (Business Case, Feasibility Assessment, Options Evaluation, Rapid Prototyping if any, Decision making), Project Management, Compliance and Regulatory requirements (PIA, STRA etc), Team Review of deliverables/work products, approvals and signoffs, Post-project evaluation (Project completion and closure, Project evaluation, lessons learnt).

Analysis : "Business Requirements" *means both "Functional" and "Non-Functional Requirements".* Business Requirements gathering/elicitation/capturing, Business Requirements modeling, Business Requirements documentation, Business Requirements Management (requirements prioritization, requirements re-validation etc), Team Review of requirements deliverables/work products, Compliance and Regulatory requirements (PIA, STRA etc if any), approvals and signoffs. NOTES : A "Use Case" is not a requirement, but a notation to represent a requirement in a model/diagram. An "Actor" is a not an actual user, but a notation to represent the actual user in a model/diagram. In some projects, "Technical Requirements Analysis" may also be done to some extent in the Analysis phase in addition to doing the "Business Requirements Analysis".

Design : Technical Requirements Analysis if any, Solution Options Analysis if any, Technical Architecture work if any, Data Architecture work, Application Architecture work, Design of the proposed solution, Team Review of architecture and design deliverables/work products, approvals and signoffs. NOTE : In case of COTS, Open-Source and SaaS implementations : Architecture/Design work may only involve any customization of metadata, user experience artefacts, report formats etc.

Implementation : Coding/Building, Testing (unit testing, system testing, user acceptance testing (UAT) and performance/load/stress testing), UAT sign off, production rollout. NOTE : In case of COTS, Open-Source and SaaS implementations : The implementation work will vary depending on the project scope.

Build : Coding, unit testing and unit-bugs fixing, System testing and system-bugs fixing, Integration/interface testing and Integration bug-fixing, approvals and signoffs for UAT migration. NOTE : "Build" phase may not be entirely applicable to COTS, Open-Source and SaaS situations.

User Acceptance : UAT testing , UAT feedback fixing, UAT signoff.

Performance Testing : Performance testing (Performance/Load/Stress Testing), Performance issues fixing, Performance Testing signoff. NOTE : Scope of "Performance Testing" may vary for COTS, Open-Source and SaaS situations.

Production Rollout : Production testing if any, production signoff.